National Technical University of Athens

School of Electrical and Computer Engineering Semester 7

Final project in the course Multimedia Technology

Application description

In the context of the project a Task Management System will be implemented. The application will allow the user to create, edit, and monitor the available tasks. In addition, the user will be able to manage multiple tasks, set priorities and deadlines, and receive reminders for upcoming tasks.

A.1. Design and implementation of logic (40%)

The following describes the features that the application should provide to the user regarding the creation and management of tasks and reminders.

Add, modify, and delete tasks

The user will be able to create new tasks. The relevant information for a task should include: title, description, category, priority, completion deadline, and status. For simplicity, the completion deadline for a task will not include a time, but will be set at the day level (e.g. 12/14/24)

The status of a job will always be one of the following: "*Open*", *"In Progress"*, *"Postponed"*, *"Completed"*, and *"Delayed".* For each new task the default status will be *"Open".* When initializing the application, tasks that are not *"Completed"* and have passed the completion deadline should be detected and the status should automatically change to *"Delayed".*

The user will be able to modify all the elements of a task as well as proceed to delete tasks. In the case of deletions, care must be taken to ensure that the possible reminders set for the task to be deleted are properly updated.

Add, modify, and delete a category

The user will be able to define new categories by giving them a name. In addition, they will be able to modify the name of a category. For simplicity we assume that there are no subcategories.

It will also be able to delete a category along with automatically deleting all the tasks belonging to it. In this case the reminders for the deleted tasks should be updated appropriately.

Add, modify, and delete priority

The application should include a default priority level named *"Default"*. The user will be able to define new priority levels by giving the relevant name. In addition, it will be possible to modify the name of a priority level as well as delete priority levels.

The change name and delete functions will apply to all priority levels except the default. Also, when a priority level is deleted then all tasks belonging to the relevant level shall automatically be assigned the default priority level.

Additional functions

Set and manage reminders: the user will be able to create reminders for tasks. A reminder will always be associated with a task, and multiple reminders can be set for a task. If a task has a status of *"Completed"* there will be no possibility to set reminders, and when the user changes the status of a task to *"Completed"* then the application will automatically delete any reminders related to that task.

The application should support the following types of reminders: (i) one day before the deadline, (ii) one week before the deadline, (iii) one month before the deadline, (iv) a specific date set by the user. Appropriate checks should be implemented to ensure that the selected reminder type is meaningful based on the deadline for completion of the task and that in case of an issue the user is informed with the corresponding message. Finally, the user will be able to modify and delete reminders.

Search for jobs. The user will be able to search for tasks based on any combination of the following criteria: title, category, and priority.

A.2. Storage and retrieval of application information (10%)

A solution based on the use of files containing data in JSON format will be used to store and retrieve the application information.

JSON (JavaScript Object Notation) is a data representation format widely used for storing and transferring data. It is easy to read by humans and, at the same time, understandable by several programming languages. JSON follows a simple structure using text to represent data in a key-value format. More information can be found in the workshop slides in the *"Java Networking & JSON"* section.

First you need to decide and define your own organization (data schema) for the JSON data and the set of files that will be used to store the application data. The application data files should be stored in a folder named ***"medialab".***

Next you need to implement through the appropriate classes the methods that will allow you to retrieve the information that the files have and initialize the appropriate objects in your application and refresh the files so that the overall state can be maintained between intermittent runs of the application.

We then describe the logic to be implemented to retrieve and update the application data.

- Application initialization: You should be retrieving all the information in the JSON files and initializing the corresponding objects in your application at the same time.
- Execute application: The application will use the state information retrieved in program memory during initialization. All operations related to tasks and reminders managed by the application will be executed based on the information in program memory.
- Application termination: JSON files with system state information will be refreshed exclusively before application termination. The implementation shall store in the corresponding JSON files the total state of the application at the time of termination.

A.3. Graphical interface creation (30%)

You need to design and implement the appropriate graphical user interface (Graphical User Interface - GUI) using the JavaFX framework [1][2].

Note: The following are the basic specifications for the graphical interface, for all the details of the final implementation you can make any choices you want regarding the appearance and the general user interaction with the application, without any impact on the final score. For example, you can choose a simple visualization for the various elements or combine various features from JavaFX to create an effect that corresponds to a modern application. In any case, there is no need to make this part of the task complicated.

Initially when starting the application if there are any tasks that are in *"Delayed"* status the user should be informed with an appropriate popup window about the number of overdue tasks.

When creating the GUI you should follow the general instructions below:

- Create the main "window" of the application with the title "MediaLab Assistant" and set the appropriate dimensions.
- Split the window into two main parts.
- At the top of the screen, aggregated information will be displayed that will be updated accordingly based on the user's actions. The information includes (i) total number of tasks regardless of status, (ii) number of tasks with status *"Completed"*, (iii) number of tasks with status *"Delayed"*, and (iv) number of tasks with a completion deadline within 7 days.
- In the other part of the screen the various functions that the GUI must support should be implemented. There is complete freedom on how to implement these functions, however, you should ensure that the GUI information is updated appropriately according to the user's actions.

The GUI should support the following functions:

- Task management: the application must present the available tasks by category. It should also be possible for the user to define a new task, modify a task and delete a task. The implementation must be in accordance with the corresponding logic from section A.1.

- Category management: the application should present the list of categories. Furthermore, the user should be able to define new categories, change the name of a category and delete categories. The implementation should be done according to the corresponding logic from section A.1.
- Priority level management: the application should present the available priority levels. The user should be able to modify the name of the priority levels and delete priority levels. The implementation should be done according to the corresponding logic from section A.1.
- Reminder management: the application should show all active reminders. The user should be able to modify a reminder as well as delete reminders. The implementation should be done according to the corresponding logic from section A.1.
- Search for tasks: there should be a corresponding form that allows searching for tasks based on the specifications from section A.1.The results should include: title, priority level, category and deadline for completion.

Finally, your GUI implementation should ensure that when the application is terminated, the system information in the relevant files is updated according to the procedure described in section A.2.

A.4. Other receivables (20%)

- H Implementation will should be follow the principles design principles object-oriented programming (OOP design principles).
- In a class of your choice, each public method it contains should be documented according to the javadoc tool specification [3].

Note: For anything that is not clear from the pronunciation you can make your own assumptions and guesses. The pronunciation outlines the basic requirements for the implementation, however you can make your own design assumptions trying to make the implementation more realistic without making the implementation complex.

Deliverables

- The project (of the IDE of your choice) with the implementation code of the application.
- A <u>short (max 3 pages)</u> report containing a general description of the implementation design and a description of your design regarding the structure of the information in the different JSON files. You will also mention any functionality you have not implemented and any assumptions you have made. Under no circumstances should you include code fragments.

References

[1] https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm
[2] https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
[3] https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html